CLAIMS:

1. A processor-based method performed by software emulating an instruction processor, the method comprising:

processing read instructions with an emulated processor executing within an emulation environment to output independent read requests via an operand interface and an op-code interface of the emulated processor;

independently comparing op-code reference data and operand reference data to operands and op-codes received in response to the read requests; and

recording results of the independent comparisons.

2. The method of claim 1, wherein independently comparing further comprises:

storing the op-code reference data and the operand reference data within a set of data memories of the emulated instruction processor;

maintaining within the emulated instruction processor an operand data pointer to address the operand reference data and an op-code pointer to address the op-code reference data; and

independently accessing the operand reference data with the operand data pointer and the op-code reference data with the op-code data pointer during processing of the read instructions to verify the received op-codes and the received operands.

3. The method of claim 2, further comprising:

when processing the read instructions, determining whether each of the read instructions required an operand read request or an op-code read request; and

independently updating the op-code reference pointer or the operand reference pointer based on the determination.

4. The method of claim 2, wherein the read instructions form part of an instruction stream executed by the emulated instruction processor, the method further comprising:

processing a flow control instruction of the instruction stream with the emulated instruction processor; and

upon processing the flow control instruction, synchronizing the op-code reference pointer and the operand reference pointer to respectively address a portion of the op-code reference data and a portion of the operand reference data associated with a target address of the flow control instruction.

5. The method of claim 2, compiling test software to output the operand reference data, the op-code reference data, and the instruction stream.

6. The method of claim 2, wherein independently comparing further comprises:

latching the op-code reference data within a first latch within the emulated instruction processor; and

comparing the latched op-code referenced data and the received op-codes with a comparator to produce the results.

7. The method of claim 2, wherein independently comparing further comprises:

latching the operand reference data within a first latch within the emulated instruction processor; and

comparing the latched operand referenced data and the received operand with a comparator to produce the results.

8. The method of claim 1, further comprising:

storing write data within a data memory of the emulated instruction processor;

maintaining within the emulated instruction processor a write data pointer to address the write data; and

processing a write instruction with the emulated processor to output a write request via a data interface of the emulated processor, wherein the write request comprises a portion of the write data referenced by the write pointer.

9. The method of claim 1, further comprising generating a report based on the results, wherein the report identifies any of the received op-codes that do not match the op-code reference data and any of the received operands that do not match the operand reference data.

10. The method of claim 1, wherein recording results comprises storing addresses associated with received operands and the op-codes, copies of the received operands or op-codes, copies of the reference operands and the reference op-codes, or copies of the instruction.

11. The method of claim 1, wherein recording the results comprises recording the results within a register within the emulated instruction processor.

12. The method of claim 1, further comprising applying bit masks to at least a portion of a result of the independent comparisons of the op-code reference data and the operand reference data to the op-code and the operand received in response to the read requests.

13. The method of claim 1, further comprising:

storing the addresses of a small number of received operands and received op-codes within a cache within the emulated instruction processor; and

selectively enabling and disabling access to the cache when executing subsequent read instructions and waiting to output read requests via the operand interface and the op-code interface when the read instructions request operands and op-codes are invalidated within the cache based on a configurable option.

14. A processor-based system for emulating an instruction processor comprising:

a computing system to provide an emulation environment; and

software executing within the emulation environment to emulate an instruction processor having an operand interface and an op-code interface,

wherein the software emulates the instruction processor by processing read instructions and outputting corresponding read requests on the operand interface or the op-

code interface, and independently comparing op-code reference data and operand reference data to operands and op-codes received from the operand interface and op-code interface in response to the read requests.

15. The system of claim 14, wherein the software emulates the instruction processor by recording results of the independent comparisons within a register of the emulated instruction processor.

16. The system of claim 14, wherein the emulated instruction processor comprises:
a first data memory to store the op-code reference data; and
a second data memory to store the operand reference data.

17. The system of claim 16, wherein the emulated instruction processor comprises:
a control unit that maintains an operand data pointer to address the operand reference data within the first data memory and an op-code pointer to address the op-code reference data within the second data memory,
wherein the control unit independently accesses the operand reference data with the operand data pointer and the op-code reference data with the op-code data pointer during processing of the read instructions to verify the received op-codes and the received operands.

18. The system of claim 17, wherein upon processing the read instructions, the control unit determines whether each of the read instructions required an operand read request or an op-code read request, and independently updates the op-code reference pointer or the operand reference pointer based on the determination.

19. The system of claim 17, wherein the read instructions form part of an instruction stream executed by the emulated instruction processor, and upon processing a flow control instruction of the instruction stream, the control unit synchronizes the op-code reference pointer and the operand reference pointer to respectively address a portion of the op-code

reference data and a portion of the operand reference data associated with a target address of the flow control instruction.

20. The system of claim 19, further comprising a compiler executing on the computing system to compile test software to output the operand reference data, the op-code reference data, and the instruction stream for execution by the emulated instruction processor.

21. The system of claim 19, wherein the emulated instruction processor further comprises:

a latch to latch the op-code reference data from the first data memory; and

a comparator to compare the latched op-code referenced data and the received op-codes.

22. The system of claim 19, wherein the emulated instruction processor further comprises:

a latch to latch the operand reference data from the second data memory; and

a comparator to compare the latched operand referenced data and the received operands.

23. The system of claim 14, wherein the emulated instruction processor further comprises:

a data memory to store write data; and

a control unit to maintain a write data pointer to address the write data, wherein the control unit processes a write instruction to output a write request via a data interface of the emulated processor, and further wherein the control unit generates the write request to comprises a portion of the write data referenced by the write pointer.

24. The system of claim 14, further comprising emulation control software executing on the computing system to generate a report that presents/identifies any of the received op-

codes that do not match the op-code reference data and any of the received operands that do not match the operand reference data.

25. The system of claim 14, wherein the emulated instruction processor further comprises:

a set of memories to store bit masks; and

bit masks to compare results of the comparison of the received operands and the received op-codes to the reference operands and the reference op-codes to mask portions of these comparisons.

26. The system of claim 14, wherein the emulated instruction processor further comprises a cache to store the addresses of the received operands and received op-codes, wherein the emulated instruction processor selectively waits to issue read requests for subsequent read instructions via the operand interface and the op-code interface until the read instructions request operands and op-codes become invalidated within this cache based on a configurable option.

27. A processor-based system for emulating an instruction processor comprising:

compiling means for compiling test software to produce operand reference data, op-code reference data, and an instruction stream having read instructions; and

emulating means for emulating an instruction processor having an operand interface and an op-code interface,

wherein the emulating means comprises:

controlling means for controlling the emulated instruction processor to process the read instructions and output corresponding read requests on the operand interface or the op-code interface,

receiving means for receiving operands and op-codes from the operand interface and op-code interface in response to the read requests, and

comparing means for independently comparing the op-code reference data and the operand reference data to the received.

28. The system of claim 27, wherein the emulating means further comprises:

a first storing means for storing the op-code reference data; and

a second storing means for storing the operand reference data.

29. The system of claim 27, wherein the controlling means comprises:

first referencing means for addressing the operand reference data within the first storing means; and

second referencing means for addressing the op-code reference data within the second storing means.

30. The system of claim 27, further comprising reporting means for generating a report that presents and identifies any of the received op-codes that do not match the op-code reference data and any of the received operands that do not match the operand reference data.